DESIGN OF PATTREN RECOGNIZATION BY NEURAL NETWORKS

Dara Eshwar[1], Dr. M.V. Ramanamurthy[2]
[1]Research Scholar, CMJ University, Shillong, Meghalaya, India
[2]Professor, Osmania University, Hyderabad, India

ABSTRACT

The design of a pattern recognition system essentially involves the following three aspects such as data acquisition and preprocessing, data representation, and decision making. The problem domain dictates the choice of sensor(s), preprocessing technique, representation scheme, and the decision-making model. It is generally agreed that a well-defined and sufficiently constrained recognition problem (small intra-class variations and large interclass variations) will lead to a compact pattern representation and a simple decision-making strategy. Learning from a set of examples (training set) is an important and desired attribute of most pattern recognition systems. The four best known approaches for pattern recognition are template matching, statistical classification, syntactic or structural matching, and neural networks. These models are not necessarily independent and sometimes the same pattern recognition method exists with different interpretations. Attempts have been made to design hybrid systems involving multiple models.

Keywords: Pattern recognition, Preprocessing technique, Representation scheme, Decision-making model.

1. INTRODUCTION

Automatic (machine) recognition, description, classification, and grouping of patterns are important problems in a variety of engineering and scientific disciplines [1] such as biology, psychology, medicine, marketing, computer vision, artificial intelligence, and remote sensing. But what is a pattern Watanabe [2] defines a pattern as opposite of a chaos; it is an entity, vaguely defined, that could be given a name. For example, a pattern could be a fingerprint image, a handwritten cursive word, a human face, or a speech signal. Given a pattern, its recognition/classification may consist of one of the following two tasks: (1) supervised classification (e.g., discriminate analysis) in which the input pattern is identified as a member of a predefined class, (2) unsupervised classification (e.g., clustering) in which the pattern is assigned to a hitherto unknown class. Note that the recognition problem here is being posed as a classification or categorization task, where the classes are either defined by the system designer (in supervised classification) or are learned based on the similarity of patterns (in unsupervised classification). Interest in the area of pattern recognition has been renewed recently due to emerging applications which are not only challenging but also computationally more demanding. These applications include data mining (identifying a pattern, e.g., correlation, or an outlier in millions of multidimensional patterns), document classification (efficiently searching text Documents), financial forecasting, organization and retrieval of multimedia databases, and biometrics (Personal identification based on various physical attributes such as face and fingerprints).

Whitley [3] has identified a novel application of pattern recognition, called affective computing which will give a computer the ability to recognize and express emotions, to respond intelligently to human emotion, and to employ mechanisms of emotion that contribute to rational decision making. A common characteristic of a number of these applications is that the available features (typically, in the thousands) are not usually suggested by domain experts but must be extracted and optimized by data-driven procedures.

The design of a pattern recognition system essentially involves the following three aspects: 1) data acquisition and preprocessing, 2) data representation, and 3) decision making. The problem domain dictates the choice of sensor(s), preprocessing technique, representation scheme, and the decision-making model. It is generally agreed that a well-defined and sufficiently constrained recognition problem (small intra-class variations and large interclass variations) will lead to a compact pattern representation and a simple decision-making strategy. Learning from a set of examples (training set) is an important and desired attribute of most pattern recognition systems. The four best known approaches for pattern recognition are: 1) template matching, 2) statistical classification, 3) syntactic or structural matching, and 4) neural networks. These models are not necessarily independent and sometimes the same pattern recognition method exists with different interpretations. Attempts have been made to design hybrid systems involving multiple models [4]. A brief description and comparison of these approaches is given below:

2. Pattern Recognition Methods

The design of a pattern recognition system essentially involves the following three aspects: 1) data acquisition and preprocessing, 2) data representation, and 3) decision making. The problem domain dictates the choice of sensor(s), preprocessing techniques, representation scheme, and the decision-making model. Learning from a set of examples (training set) is an important and desired attribute of most pattern recognition systems.

2.1. Template Matching

One of the simplest and earliest approaches to pattern recognition is based on template matching. Matching is a generic operation in pattern recognition, which is used to determine the similarity between two entities (points, curves, or shapes) of the same type. In template matching, a template (typically, a 2D shape) or a prototype of the pattern to be recognized is available. The pattern to be recognized is matched against the stored template while considering all allowable pose (translation and rotation) and scale changes. The similarity measure, often a correlation, may be optimized based on the available training set. Often, the template itself is learned from the training set. Template matching is computationally demanding, but the availability of faster processors has now made this approach more feasible. The rigid template matching mentioned above, while effective in some application domains, has several disadvantages. For instance, it would fail if the patterns were distorted due to the imaging process, viewpoint change, or large intraclass variations among the patterns. Deformable template models [5] or rubber sheet deformations [6] can be used to match patterns when the deformation cannot be easily explained or modeled directly.

2.2. Statistical Approach

In the statistical approach, each pattern is represented in terms of d features or measurements and is viewed as a point in a d-dimensional space. The goal is to choose those features that allow pattern vectors belonging to different categories to occupy compact and disjoint regions in a dimensional feature space. The effectiveness of the representation space (feature set) is determined by how well patterns from different classes can be separated. Given a set of training patterns from each class, the objective is to establish decision boundaries in the feature space which separate patterns belonging to different classes. In the statistical decision theoretic approach, the decision boundaries are determined by the probability distributions of the patterns belonging to each class, which must either be specified or learned [7], [8]. One can also take a discriminant analysis-based approach to classification: First a parametric form of the decision boundary (e.g., linear or quadratic) is specified; then the "best" decision boundary of the specified form is found based on the classification of training patterns. Such boundaries can be constructed using, for example, a mean squared error criterion. The direct boundary

construction approaches are supported by Vapnik's philosophy [9]: "If you possess a restricted amount of information for solving some problem, try to solve the problem directly and never solve a more general problem as an intermediate step. It is possible that the available information is sufficient for a direct solution but is insufficient for solving a more general intermediate problem."

2.3. Syntactic Approach

In many recognition problems involving complex patterns, it is more appropriate to adopt a hierarchical perspective where a pattern is viewed as being composed of simple sub patterns which are themselves built from yet simpler sub patterns [10], [11]. The simplest/elementary sub patterns to be recognized are called primitives and the given complex pattern is represented in terms of the interrelationships between these primitives. In syntactic pattern recognition, a formal analogy is drawn between the structure of patterns and the syntax of a language. The patterns are viewed as sentences belonging to a language, primitives are viewed as the alphabet of the language, and the sentences are generated according to a grammar. Thus, a large collection of complex patterns can be described by a small number of primitives and grammatical rules. The grammar for each pattern class must be inferred from the available training samples. Structural pattern recognition is intuitively appealing because, in addition to classification, this approach also provides a description of how the given pattern is constructed from the primitives. This paradigm has been used in situations where the patterns have a definite structure which can be captured in terms of a set of rules, such as EKG waveforms, textured images, and shape analysis of contours [10]. The implementation of a syntactic approach, however, leads to many difficulties which primarily have to do with the segmentation of noisy patterns (to detect the primitives) and the inference of the grammar from training data. Fu [10] introduced the notion of attributed grammars which unifies syntactic and statistical pattern recognition. The syntactic approach may yield a combinatorial explosion of possibilities to be investigated, demanding large training sets and very large computational efforts [12].

Pattern recognition is the categorization of input data into identifiable classes through the extraction of significant attributes of the data from irrelevant background details. A pattern class is a category determined by some common attributes. Therefore, a pattern is the description of a category member representing a pattern class.

Pattern recognition, as a subject, spans several scientific disciplines, uniting them in search for a solution to the common problem of recognizing members of a given class. Often these are problems that many humans solve in a seemingly effortless fashion. However, their solution using computers has, in many cases, proved to be immediately difficult. In order to have the best opportunity of developing effective solutions, it is important to adopt a principled approach based on sound theoretical concepts.

A pattern could be a fingerprint image, a handwritten cursive word, a human face, or a speech signal. Given a pattern, its recognition/classification may consist of the following two tasks 1) supervised classification (e.g., discriminant analysis) in which the input pattern is identified as a member of a predefined class, 2) unsupervised classification (e.g., clustering) in which the pattern is assigned to a hitherto unknown class. Interest in the area of pattern recognition has been reviewed recently due to emerging applications, which are not only challenging but also computationally more demanding. These applications include data mining (identifying a "pattern" e.g., correlation, or an outlier in millions of multidimensional patterns), document classification (efficiently searching text documents), financial forecasting, organization and retrieval of multimedia databases, and biometrics. Various works has identified a novel application of pattern recognition, called affective computing, which will

give a computer the ability to recognize and express emotions, to respond intelligently to human emotion, and to employ mechanisms of emotion that contribute to relation decision making. A common characteristic of a number of these applications is that the features are not usually suggested by domain experts but must be extracted and optimized by data-driven procedures.

3. Neural Networks

As discussed in the previous chapter Neural network, massively parallel computing systems with large number of processors interconnected with each other, models attempt to use some organizational principles (such as learning, generalization, adaptively, fault tolerance, computation and distributed representation) in a network of weighted directed graphs. The main characteristics of neural networks are that they can learn complex non-linear input output relationships, use sequential training procedures and adapt themselves to the data.

By suitable choice of architecture for a feed forward network and parameters of a feedback network, it is possible to perform various pattern recognition tasks like pattern association, pattern classification, auto association, pattern storage etc. The simplest task is a pattern association task, which can be realized by a two-layer feed forward network with linear processing units. This is called a linear associative network. The objective in pattern association is to design a network that can represent the association in the pairs of vectors (l 1, 2,....L, al bl) through a set of weights determined by a learning law. The given set of input-output pattern pairs is called training data. The input patterns are generated synthetically, like machine printed characters. The input patterns used for recall may be corrupted by external noise. The following vector and matrix notations are used for the analysis of a linear associative network.

Input vector

$$\mathbf{a}_l = [a_{l1}, a_{l2}, \ldots, a_{lM}]^T$$

Activation vector of input layer

$$\mathbf{x} = [x_1, x_2, \ldots, x_M]^T$$

Activation vector of output layer

$$\mathbf{y} = [y_1, y_2, \ldots, y_N]^T$$

Output vector

$$\mathbf{b}_l = [b_{l1}, b_{l2}, \ldots, b_{lN}]^T$$

Input Matrix $A = [\mathbf{a}_1, \mathbf{a}_2, \ldots \mathbf{a}_L]$ **of M x L**

Output Matrix

$$B = [\mathbf{b}_1, \mathbf{b}_2, \ldots \mathbf{b}_L] \text{ of N X L}$$

Weight Matrix $W = [\mathbf{w}_1, \mathbf{w}_2, \ldots \mathbf{w}_N]$ **of N X M**

Weight vector for jth unit of output layer

$$\mathbf{w}_j = [w_{j1}, w_{j2}, \ldots, w_{jM}]^T$$

The network consists of a set of weights connecting two layers of the processing units as shown in figure. The output function of each unit in these layers is linear. Each output unit receives inputs from the M input units corresponding to the M-dimensional input vectors. The number of

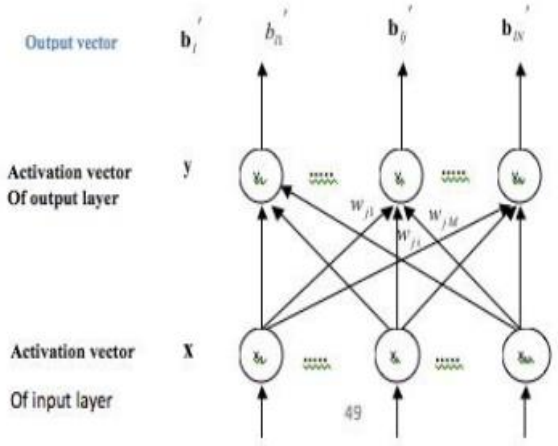

Fig. 1: Linear Associative Network

output units correspond to the dimensionality of the output vectors. Due to the linearity of the output function, the activation value (x_i) and the signal values of the units in the input layer are the same as the input data values a_{li}. The activation value of the j^{th} unit in the output layer is given by

$$y_j = \sum_{i=1}^{M} w_{ji} a_{li} = \mathbf{w}_j^T \mathbf{a}_l, \; j = 1,2,\ldots,N$$

Here the objective is to determine a set of weights $\{w_{ji}\}$ in such a way that the actual output ′ blj is equal to the desired output b l_j for all the given L pattern pairs. The weights are determined by using the criterion that the total mean squared error between the desired output is to be minimized. The total error E(W) over all the L input-output pattern pairs is given by

$$E(W) = \frac{1}{L} \sum_{l=1}^{L} \sum_{j=1}^{N} (b_{lj} - b'_{lj})^2 = \frac{1}{L} \sum_{l=1}^{L} \left\| \mathbf{b}_l - W\mathbf{a}_l \right\|^2$$

We can also write this as,

$$E(W) = \frac{1}{L} \|B - WA\|^2$$

Similarly using feedback network consisting of linear processing units, we can realize the auto association task of pattern recognition. Another pattern recognition task, which can be realized by neural networks, is pattern storage and recalling. The objective in pattern storage task is to store a given set of patterns, so that any of them can be recalled exactly when an approximate version of the corresponding pattern is presented to the network. For this purpose, the features and their spatial relations in the patterns need to be stored. The pattern recall should take place even when the features and their spatial relations are slightly disturbed due to noise and distortion or due to natural variation of the pattern generating process. Sometimes the data itself is stored through the weights, as in the

case of binary patterns. In this case the approximation can be measured in terms of some distance, like Hamming distance, between the patterns.

A feedback network consisting of processing units with nonlinear output functions generally accomplishes pattern storage. The output of the processing units at any instant of time defines the output state of the network at that instant. The state of the network at successive instants of time, i.e., the trajectory of the state, is determined by the activation dynamics model used for the network. Recall of the stored pattern involves starting at some initial state of the network depending on the input and applying the activation dynamics until the trajectory reaches an equilibrium state. The final equilibrium state is the stored pattern resulting from the network for the given input.

Associated with each output state is an energy, which depends on the network parameters like the weight's bias, besides the state of the network. The energy as a function of the state of the network corresponds like an energy landscape. The shape of the energy landscape is determined by the network parameters and states. The feedback among the units and the nonlinear processing in the units may create basins of attraction in the energy landscape, when weights satisfy certain constraints. Figure 3.2 shows energy landscapes as a function of output for the two cases of with and without the basins of attraction

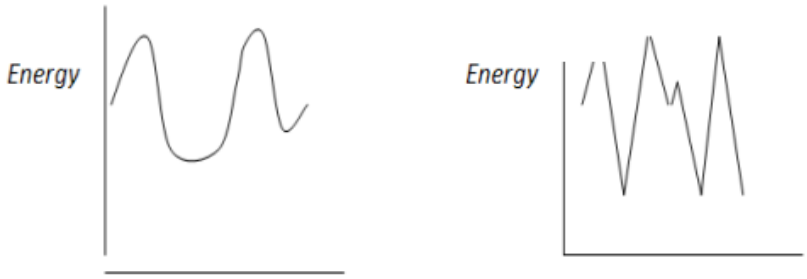

Fig. 2: Energy landscapes (a) with basins of attraction and (b) without basins of attraction.

In the latter case the energy fluctuates quickly and randomly from one state to another as shown in Figure 2b. But in the energy landscape with basins of attraction as in Figure 2a, the states around the stable state correspond to small deviations from the stable state. The deviation can be measured in some suitable distance measure, such as Hamming distance for binary patterns. The Hamming distance between two binary patterns each of length N is defined as the number of bit positions in which patterns differ. Thus, the states closer to the stable states correspond to patterns with smaller Hamming distance.

The basins of attraction in the energy landscape tend to be the regions of stable equilibrium states [33]. If there is a fixed state in each of the basins where the energy is minimum, then the state corresponds to a fixed point of equilibrium.

The existence of the basins of attraction or regions of equilibrium states is exploited for the pattern recognition task. The fixed points in these regions correspond to the states of the energy minima, and they are used to store the desired patterns. These stored patterns can be recalled even with approximate patterns as inputs. An erroneous pattern is more likely to be closer to the corresponding true pattern than to the other stored patterns according to some distance measure. The number of patterns that can be stored is called the capacity of the network. It is possible to estimate the capacity of the network and also the average probability of error in recall. The probability of error in recall can be reduced by adjusting the weights in such a way that the resulting energy landscape is matched to

the probability distribution of the desired patterns. Typically, the capacity of a fully connected network is of the order of N, the number of processing units. Although there are 2 N different states for a network with binary state units, the network can be used to store only of the order of N binary patterns, as there will be as many fixed points as possible or energy minima in the energy landscape.

If the number of patterns is more than the number of basins of attraction, then the storage problem becomes hard problem, in the sense that the patterns cannot be stored in the given network. On the other hand, if the number of patterns is less than the number of basins of attraction, then there will be the so-called false well or false minima due to the additional basins of attraction. During recall, it is likely that the state of the network may settle in a false well. The recalled pattern corresponding to the false well may not be the desired pattern, thus resulting in an error in the recall. Hopfield model is a fully connected model of a feedback network for the pattern storage. This model is used in the present research.

Errors in pattern due to false minima can be reduced significantly if initially the desired patterns are stored at the lowest energy minima of a network, during training. The error can be reduced further by using suitable activation dynamics. The activation dynamics is modified so that the network can also move to a state of higher energy value initially, and then to the nearest deep energy minima. This way error in recall due to false minima can be reduced.

It is possible to realize a transition to a higher energy state from a lower energy state by using a stochastic update in each unit instead of the deterministic update of the output function as in Hopfield model. In a stochastic update the activation value of a unit does not decide the next output state of the unit directly using the output function f (x), instead, the update is expressed in probabilistic terms, like the probability of firing by the unit being greater than 0.5 if the activation exceeds the threshold.

Loading of the pattern environment is accomplished in a feedback network with stochastic units using Boltzmann machine A fully connected network consisting of both hidden and visible units and operating asynchronously with stochastic update for each unit is called Boltzmann machine. Boltzmann learning law is too slow for implementation in any practical situations involving pattern environment storage. For practical implementation, an approximation in the form mean-field annealing is used.

3.2. Hopfield neural network and pattern storage

Our objective is to store a given set of patterns so that any one of the patterns can be recalled exactly when an approximate or corrupted version, due to noise and distortion, of the pattern is presented to the network. What is stored in practice is the information in the pattern itself. The approximation is measured in the terms of some distance, like Hamming distance. The distance feature is automatically realized through the threshold (binary) feature of the output function of the processing unit. A feedback network consisting of non-linear processing units accomplishes the pattern storage.

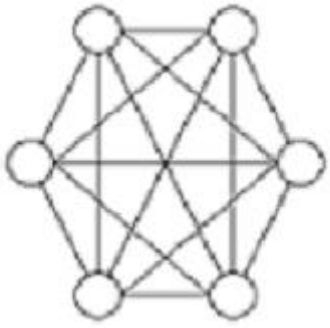

Fig. 3: Hopfield network structure with six units.

Popularized by Nobel Laureate John Hopfield, the Hopfield model is a fully connected feedback network with symmetric weights (i.e. $w_{ij} = w_{ji}$), which possess a rich class of dynamics characterized by the existence of several stable states each with its own basin of attraction These basins of attraction in the energy landscape tend to be the regions of stable equilibrium states The number of basins of attraction in the energy 243 landscape depends only on the network i.e. the number of processing units and their interconnection strengths (weights). When the number of patterns to be stored is less than the number of basins of attraction, i.e., stable state, then there will be spurious stable states, which do not correspond to any desired patterns. In such a case, when the network is presented with an approximate pattern for recall, the activation dynamics may eventually lead to a stable state that may correspond to one of the spurious state or a false energy minimum, or to one of the stable states corresponding to some other pattern. In the latter case, there will be an undetected error in the recall. The average probability of error depends on the energy values of the stable states corresponding to the desired patterns, and the relative locations of these states in the state space, measured in terms of some distance criterion.

$$s_i = f(x\ ) = \mathrm{sgn}(x_i)$$

and

$$x_i = \sum w_{ij} s_i - \theta_i$$

whereθ_i is the threshold for the unit i. We will assume $\theta_i = 0$ for convenience. The state of each unit is either +1 or –1 at any given instant of time. Due to feedback, the state of a unit depends on the states of the other units. The updating of the state of a unit can be done synchronously or asynchronously. In the synchronous update all the units are simultaneously updated at each time instant, assuming that the state of the network is frozen until update is made for all the units. In the asynchronous update a unit is selected at random, and its new state is computed. Another unit is selected at random, and its state is updated using the current state of the network. The updating using the random choice of a unit is continued until no further change in the state takes places for all the units. That is, the state at time (t+1) is the same as the state at time t for all the units. That is,

$$s_i(t+1) = s_i(t), \qquad \text{for all } i$$

In this situation, we can say that the network activation dynamics reached a stable state. We assume asynchronous update for future because asynchronous update ensures that the next state is at most unit Hamming distance from the current state. If the network is to store a pattern $a = (a_1, a_2, \ldots, a_N)^T$, then in a stable state we must have the updated state value to be the same as the current state value. That is,

$$\mathrm{sgn}\left(\sum_{j=1}^{N} w_{ij} a_j\right) = a_i, \qquad \text{for all } i$$

This can happen if ( ) $w_{ij} = (1/N)\, a_i\, a_j$, because

$$\sum_{j=1}^{N} w_{ij} a_j = \frac{1}{N} \sum_{j=1}^{N} a_i a_j a_j = \frac{a_i}{N} \sum_{j=1}^{N} a_j^2 = a_i$$

where $a^2_j = 1$ for bipolar (± 1) states.

$$w_{ij} = \frac{1}{N}\sum_{l=1}^{L} a_{li} a_{lj}$$

then the state a_k will be stable if

$$\operatorname{sgn}\left(\frac{1}{N}\sum_{j=1}^{N}\sum_{l=1}^{L} a_{li} a_{lj} a_{kj}\right) = a_{ki}, \text{for all } i$$

Taking out the $l = k$ term in the summation and simplifying it by using $a^2_{kj} =$, we get,

$$\operatorname{sgn}\left(a_{ki} + \frac{1}{N}\sum_{j=1}^{N}\sum_{l\neq k} a_{li} a_{lj} a_{kj}\right) = a_{ki}, \text{for all } i$$

Since $a_{ki} = \pm$, the above is true for all a_{ki}, provided the cross term in above equation does not change the sign of a_{ki} plus the cross term.

Let the network consist of N fully connected units with each unit having hard-limiting bipolar threshold output function. Let a_l , l = 1,2,...,L be the vectors to be stored. The vectors (a_l) are assumed to have bipolar components, i.e., $a_{li} = \pm 1$, $i = 1,2,...,N$.

1. Assign the connection weights

$$w_{ij} = \frac{1}{N}\sum_{l=1}^{L} a_{li} a_{lj}, \text{ for } i \neq j$$

$$= 0, \quad \text{for } i = j \quad , 1 \le i, j \le N$$

2. Initialize the network output with the given unknown input pattern a

$$s_i(0) = a_i, \text{ for } i=1,2,...,N$$

where $s_i(0)$ is the output of the unit i at time $t = 0$

3. Iterate until convergence

Fig. 4: Hopfield network algorithm for store and recall a bipolar pattern.

3.2.1 Energy analysis of Hopfield Network Discrete Hopfield model

Associated with each state of the network, Hopfield proposed an energy function whose values always either reduces or remains the same as the state of the network changes. Assuming the threshold value of the unit i to be θ_i, the energy function is given by [33]

$$V(s) = V = -\frac{1}{2}\sum_{i}\sum_{j} w_{ij} s_i s_j + \sum_{i} \theta_i s_i$$

The energy V(s) as a function of the states of the network describes the energy landscape in the state space. The energy landscape is determined only by the network architecture, i.e., the number of units, their output functions, threshold values, connections between units and the strengths of the connections. Hopfield has shown that for symmetric weights with no self-feedback, i.e., $w_{ij} = w_{ji}$, and with bipolar {-1, +1} or binary output functions, the dynamics of the network using the asynchronous update always leads towards energy minima at equilibrium. The states corresponding to these energy minima turn out to be stable states, which means that small perturbations around it led to unstable states. Hence the dynamics of the network takes the network back to a stable state again. It is the existence of these stable states that enables us to store patterns, one at each of these states.

Let us consider the change of state due to update of one unit, say k, at some instant. All other units remain unchanged. We can write the expressions for energy before and after the change as follows:

$$V^{old} = -\frac{1}{2}\sum_i \sum_j w_{ij} s_{i_i}^{old} s_j^{old} + \sum_i \theta_i s_i^{old}$$

$$V^{new} = -\frac{1}{2}\sum_i \sum_j w_{ij} s_{i_i}^{new} s_j^{new} + \sum_i \theta_i s_i^{new}$$

The change in energy due to update of the kth unit is given by

$$\Delta V = V^{new} - V^{old} =$$

$$-\frac{1}{2}\sum_{i\neq k} \sum_{j\neq k} w_{ij}\left(s_i^{new} s_j^{new} - s_{i_i}^{old} s_j^{old}\right) + \sum_{i\neq k} \theta_i \left(s_i^{new} - s_{i_i}^{old}\right)$$

$$=$$

$$-\frac{1}{2}\sum_i w_{ik} s_{i_i}^{new} s_k^{new} - \frac{1}{2}\sum_j w_{kj} s_{k_i}^{new} s_j^{new} + \theta_k s_{k_i}^{new}$$

$$-\frac{1}{2}\sum_i w_{ik} s_{i_i}^{new} s_k^{new} - \frac{1}{2}\sum_j w_{kj} s_{k_i}^{new} s_j^{new} + \theta_k s_{k_i}^{new}$$

$$+\frac{1}{2}\sum_i w_{ik} s_{i_i}^{old} s_k^{old} + \frac{1}{2}\sum_j w_{kj} s_{k_i}^{old} s_j^{old} + \theta_k s_{k_i}^{old}$$

Since old $s_i^{new} = s_i^{old}$ $i \neq k$, the first two terms on the right hand side of Eq. (14) will be zero. Hence,

$$\Delta V = s_{k_i}^{new}\left[-\frac{1}{2}\sum_i w_{ik} s_{i_i}^{new} - \frac{1}{2}\sum_j w_{kj} s_j^{new} + \theta_k\right]$$

$$+ s_k^{old}\left[+\frac{1}{2}\sum_i w_{ik} s_{i_i}^{old} + \frac{1}{2}\sum_j w_{kj} s_j^{old} - \theta_k\right]$$

If the weights are assumed symmetric, i.e., $w_{ij} = w_{ji}$, then we get

$$\Delta V = -s_{k_i}^{new}\left[\sum_i w_{ki} s_{i_i}^{new} - \theta_k\right] + s_k^{old}\left[\sum_i w_{ki} s_{i_i}^{old} - \theta_k\right]$$

If in addition, = 0 w_{kk} , then since $s_i^{new} = s_i^{old}$ for $i \neq k$, the terms in both the parentheses are equal. Therefore,

$$\Delta V = (s_k^{old} - s_k^{new})\left[\sum_i w_{ki} s_{i_i}^{old} - \theta_k\right]$$

The update rule for each unit k is as follows

Case A: If $\sum_i w_{ki} s_{i_i}^{old} - \theta_k > 0$, then $s_{k_i}^{new} = +1$

Case B: If $\sum_i w_{ki} s_{i_i}^{old} - \theta_k < 0$, then $s_{k_i}^{new} = -1$

Case C: If $\sum_i w_{ki} s_{i_i}^{old} - \theta_k = 0$, then $s_{k_i}^{new} = s_{k_i}^{old}$ For

case A, if $s_{k_i}^{old} = +1$, then $\Delta V = 0$, and if $s_{k_i}^{old} = -1$

, then $\Delta V \leq 0$.

For case B,

if $s_{k_i}^{old} = +1$, then $\Delta V < 0$, and if $s_{k_i}^{old} = -1$, then

$\Delta V = 0$.

For case C, irrespective of the value of $s_{k_i}^{old}$,

$\Delta V = 0$.

Thus, we have $\Delta V \leq 0$. Therefore, the energy decreases or remains the same when a unit, selected at random, is updated provided the weights are symmetric, and the self-feedback is zero. This is the energy analysis for discrete Hopfield model.

4. CONCLUSION

Till now, we are through with preliminaries of pattern recognition, neural networks, genetic algorithms and one specific model of ANN i.e., Hopfield model by and large used for storing (memorizing the patterns). In the next chapter, we will try to focus our discussion on formation of hybrid evolutionary systems by combining the genetic algorithms with Hopfield model of ANN. This combination, when applied to different pattern recognition problems, gives better results compared to ANN or GA alone.

REFERENCES

[1] Ross P E, “Flash of Genius”, Forbes, pp. 98- 104, November 1998.

[2] Watanabe S, "Pattern Recognition: Human and Mechanical", New York: Wiley, 1985.
[3] Dominic S, Das R, Whitley D, and Anderson C, "Genetic reinforcement learning for neural networks", in Proceedings of IEEE Int. Joint Conf. Neural Networks (IJCNN'91 Seattle), vol. 2, pp. 71-76,1991.
[4] Devroye L, Gyorfi L and Lugosi G, A Probabilistic Theory of Pattern Recognition, Berlin: Springer-Verlag, 1996.
[5] Kosko, B., "Neural Networks and Fuzzy Systems" Prentice-Hall India,2005.
[6] Fu K S, Syntactic Pattern Recognition and Applications, Englewood Cliffs, NJ: Prentice Hall, 1982.
[7] Desai M S, Noisy pattern retrieval using associative memories, MSEE thesis, University of Louisville, Kentucky, 1990.
[8] Stanley S M, "A Theory of Evolution above the Species Level", Proceedings of National Academy of Sciences, vol 72, pp. 646450, 1975.
[9] Bremermann H J, Rogson M, "An Evolution Type Search Method for Convex Sets", ONR Technical Report, Contract 222(85) and 3656(58), UC Berkley, 1964.
[10] Yan W, Zhu Z, and Hu R, "Hybrid genetic/BP algorithm and its application for radar target classification", in Proc. 1997 IEEE National Aerospace and Electronics Conf., NAECON. Part 2 (of 2), pp. 981-984, 1997.
[11] Yao X, "Evolving Artificial Neural Network", Proceedings of the IEEE, vol. 87, number 9, September 1999.
[12] Homaifar A and Guan S, "Training weights of neural networks by genetic algorithms and messy genetic algorithms", in Proceedings of 2nd IASTED Int. Symp. Expert Systems and Neural Networks, M. H. Hamza, Ed. Anaheim, CA: Acta, pp. 74-77, 1990.
[13] Prados D L, "New learning algorithm for training multi-layered neural networks that uses genetic-algorithm techniques", Electron. Lett., vol. 28, pp. 1560-1561, July 1992.
[14] Cordella L P, Stefano C D and Fontanella F, "Evolutionary Prototyping for Handwritten Recognition", International Journal of Pattern Recognition and Artificial Intelligence, vol 21, Number 1, pp. 157-178, 2007.